Conceptual Graph Matching Method for Reading Comprehension Tests

Zhi-Chang Zhang, Yu Zhang, Ting Liu and Sheng Li, *Non-Member, KIMICS*

***Abstract*—**Reading comprehension (RC) systems are to understand a given text and return answers in response to questions about the text. Many previous studies extract sentences that are the most similar to questions as answers. However, texts for RC tests are generally short and facts about an event or entity are often expressed in multiple sentences. The answers for some questions might be indirectly presented in the sentences having few overlapping words with the questions. This paper proposes a conceptual graph matching method towards RC tests to extract answer strings. The method first represents the text and questions as conceptual graphs, and then extracts sub-graphs for every candidate answer concept from the text graph. All candidate answer concepts will be scored and ranked according to the matching similarity between their sub-graphs and question graph. The top one will be returned as answer seed to form a concise answer string. Since the sub-graphs for candidate answer concepts are not restricted to only covering a single sentence, our approach improved the performance of answer extraction on the Remedia test data.

***Index Terms*—** Reading comprehension, Conceptual graph, Candidate answer concept, Answer extraction.

I. INTRODUCTION

Reading comprehension (RC) tests are to analyze a given text and generate/extract answers in response to questions about the text (Hirschman et al., 1999). Accurately answering RC questions requires improvements in component NLP technologies, and RC tests can serve as an important driver of fundamental NLP research. On the other hand, RC tests can also be seen as single document based question answering (QA), and the methods for question analysis and answer extraction proposed in RC tests research can also be applied into the open-domain QA systems. Fig. 1 is an example of reading comprehension test task given a text and several questions about it.

While resembling the TREC QA task, RC tests task differs from it in answer extraction methods. The RC texts are often constrained in their lengths and the target answer sentence usually occurs only once or very few times (Light et al., 2001). But in TREC QA evaluations, large scale of document collections are provided and right answers responding to the questions can occur many times in the collections. Hence the TREC QA systems could exploit the redundancy of an answer appearing in multiple documents and perhaps appearing in an easily accessible form in a subset of those documents (Clarke et al., 2001). Because of short document content, information retrieval component is not needed in RC tasks and the answer to a given question must be located in the accompanying RC text, thus deep linguistic analyses for the text and questions are needed.

Watch for Northern Lights Tonight

(CANADA, March 21,1989) - Today is the first day of spring around the world. Today is special for one other reason. If you look at the sky tonight, you may see the northern lights. For years, no one knew what caused these lights ……

1. Who can see the northern lights?

2. What causes these lights?

3. When can they be seen?

Fig. 1 A Remedia story and accompanying questions

Most previous studies in RC tests rank every sentence in the given text according to the various similarities between the question and the sentence, or discriminate every sentence in text according to several features, then return a sentence that has the largest similarity with the question as the answer sentence. Some efforts handcrafted rules to extract answer sentences or exact answer phrases. These methods work well under the condition that the answer

Manuscript received July 14, 2009 ; revised September 19, 2009. Zhichang Zhang is with the IR Lab, Computer Science and Technology School of Harbin Institute of Technology, Harbin, China (Tel: +86-0451-86413683, Email: pangzhang@gmail.com)

sentences themselves provide enough evidences necessary to answer the questions.

While previous work in RC test research explored extensively different approaches to extract answers, there are still many problems to be addressed. One of these is that the propositions about entities and events mentioned in the question cannot be covered by any single sentence in accompanying text, these propositions are often scattered over several sentences. So the sentence containing the correct answer might have very few overlapping words with the question, and the top one in ranked sentence list would not be right answer sentence if the sentences are ranked with various bag-of-words or other sentence similarities. What follows shows some examples, in which Q denotes the question, and S denotes the sentence passage from the text.

1) Q: *What was the first program shown on television?*

S: [*It was on this day in 1939 that the first program was shown.*]$_i$ [...] [*The program was about the New York World's Fair.*]$_j$

2) Q: *Where is everyone watching Nadia?*

S: [*The world is watching a young girl.*]$_i$ […] [*The girl is taking part in the Olympics.*]$_j$ […] [*The girl is named Nadia*]$_k$

3) Q: *Where is the oil spill?*

S: [*The ship was carrying oil for cars and trucks.*]$_i$ […] [*Now, 11 million gallons of oil are pouring into the sea.*]$_j$ […] [*The ship is in a place called Prince William Sound.*]$_k$

The examples above show three cases. In the first and second example, while the sentence S_i has the maximum similarity with the question, the sentence S_j actually contains the correct answer. In the third example, the answer, which is contained in the sentence S_k, is impliedly associated with the concept "oil" through the concept "ship". After all sentences in text are ranked according to their similarities with the question, the order of the answer sentence might be low in ranked list.

In this paper we will concentrate on the problem of correctly extracting the answer string under the condition that the right answer sentence may not contain all propositions about entities in question. The paper explores a *Conceptual Graph Matching Method* for scoring and ranking the candidate answer concepts for a question in reading comprehension test task.

The method first constructs the conceptual graph notations for the question and text based on syntactic and shallow semantic analysis results. For every candidate answer node in the text graph, we extract a sub-graph containing this node from the text graph, and calculate its matching score with the question graph. These scores will be seen as the ranking scores of all candidate answer nodes. Then the top one answer node will be returned and used as the seed to form the complete answer string. Since the scope of sub-graphs is not restricted to covering only the concepts in a single sentence, the method is expected to be able to solve the problem we discussed before.

The rest of this paper is organized as follows. Section 2 introduces related work in reading comprehension (RC). Section 3 describes the architecture of our reading comprehension system and answer extraction flow. Section 4 details the conceptual graph matching method for answer string extraction. Section 5 talks about the evaluation corpus and metrics of our method, and discusses experimental results on the Remedia corpus. Section 6 concludes the paper and provides some further work.

II. PAPER SIZE AND FORMMAT

There has been a long history of RC research. Early studies about RC tests were mainly in AI community. In the late 1990s the RC research in NLP was begun and the Deep Read system (Hirschman, et al., 1999) was designed in a research group of MITRE Corporation as a baseline to test what was possible with the standard stock of component technologies available at that time. The group published a corpus named Remedia for RC tests, and defined an evaluation metric named *HumSent accuracy*. The previous methods in RC tests can be divided into four classes.

Bag-of-words (BOW) methods. Hirschman et al. (1999) used BOW approach to extract answer sentence augmented with stemming, named entity (NE) recognition and pronoun resolution, which achieves 36% HumSent accuracy on Remedia test set. Charniak et al. (2000) used only the verbs in question and candidate sentences to match with BOW model, and used additional strategies for different question types to achieve 41%. While multiple sentences might have the same number of words matching with the question, Xu et al. (2004) utilized the dependency relations of the question and the sentences to rank and extract answer sentence and achieved 41%.

Rule based methods. Quarc of Riloff and Thelen (2000) is a rule-based reading comprehension system, which applied many heuristic rules to compute the similarity between the candidate sentences and the question, and achieved 39.7% HumSent accuracy in Remedia test set. Du et al. (2005) utilized the Web-derived answer patterns to select the answer sentence.

Machine learning based methods. Xu et al. (2006) utilized a maximum entropy framework to extract the sentence owning the maximum probability given a

question as answer and obtained 44.7% *HumSent* accuracy, which integrated several features including the matched words and their part-of-speech (POS), the matched grammatical relations, as well as 24 heuristic rules designed by Riloff and Thelen (2000). Ng et al. (2000) employed the C4.5 classification model to discriminate whether a candidate sentence is the answer sentence, and obtained 39.3%.

Abduction based methods. Wellner et al. (2005) incorporated grammatical representation and reasoning into the system, and achieved 35% exact answer phrase metric and 46% inexact answer phrase metric. In their system, the text was first converted into a knowledge base composed of a set of propositions or facts in logic forms representing syntactic and semantic information contained in the text. After the question is converted into a logic form query, an abductive proof procedure is used to find a solution to the query using the knowledge base. The solution to the query would include the binding of the variable associated with the answer to a particular phrase in the text.

III. SYSTEM OVERVIEW

The major components of our system and the process of answer string extraction are shown in Fig. 2. The system consists of six central modules.

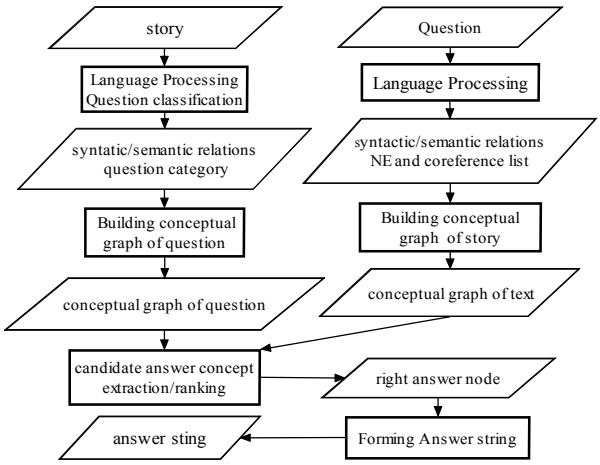

Fig. 2 Architecture of reading comprehension system based on conceptual graph

Language processing of text or question involves processes of tokenization, sentence identification, named entity identification, coreference resolution, dependency paring, shallow semantic role labeling. For questions, an additional process is to predict the category of questions with heuristic rules. The category can be used to narrow semantic scope in candidate answer string extraction.

Conceptual graph constructing is to build the graph notation for the contents in a text or question employing the information of dependency relations and semantic roles. While the dependency relations and semantic roles can only be extracted from a single sentence, the conceptual graph will connect the syntactic and semantic information from multiple sentences.

Candidate answer concept extraction and ranking involves extracting and ranking all candidate answer nodes from the text graph. If the type of a node in the text graph matches the expected answer type, the node will be extracted as a candidate answer node. For every candidate answer node, a sub-graph containing it will be extracted from the entire graph of the text. All candidate answer nodes will be ranked according to the matching scores between their sub-graphs and the question graph. The top one node will be transferred to next process stage as the best seed node for answer string formation.

Answer string formation is to extract the concise answer string from the text, using the answer seed node and its syntactic modifiers. If the extracted answer string is a reference to a named entity or early existed phrase, the named entity or the phrase will be returned as answer string.

IV. OTHERS

A. Review of conceptual graph

Conceptual graph is a formal language developed by J. F. Sowa (1979) as knowledge representation formalism. A conceptual graph is a connected graph whose nodes are either concepts or conceptual relations (or simply relations), in which these concepts and relations are connected by directed arcs.

Concepts are the representations of objects, events and states, and properties. A concept is characterized by two elements: a type which represents the set of all the occurrences of a given class (e.g., human, animal, etc.); a referent which represents a given occurrence of the class which is associated with the concept (e.g., Jack, Beijing, etc.). Using a graphical notation, a concept is represented by a rectangular box. Using a linear notation, we specify it between square brackets: [TYPE: referent].

Conceptual relation is an elementary link between two concepts. A conceptual relation is always associated with a “sink concept” (by means of an output arrow) and zero, one or several “source concepts” (by means of input arrows). Using a graphical notation, the conceptual relation is

represented by an oval box from which exits only one arrow and to which may enter several arrows. Using a linear notation it is represented between brackets and is associated with concepts by means of arrows.

A conceptual graph can be embedded into another one by use of a concept of type PROPOSITION. For instance "Mary thinks that John eats an apple" can be expressed by:

[PERSON:Mary] <- (AGNT) <- [THINK] -> (OBJ) ->
[PROPOSITION: [PERSON: John]<- (AGNT)<- [EAT] -> (OBJ) -> [APPLE]]

B. Language processing and question classification

Our conceptual graph for a text or question relies on syntactic and semantic relations in them. Every sentence in the text and the question are parsed. The results of parsing and semantic labeling, together with named entities and coreference information, are used to form the conceptual graphs of the text and question.

While the Remedia corpus used for evaluation in this paper provides the human-annotated named entities and coreference information, we made use of the Semantic Parser LTH_SRL[1], a freely available natural language parsing system, to get the dependency relations and semantic roles. Besides labeling the semantic roles of verbs according to PropBank frames, LTH_SRL could also label the semantic roles of some special nouns under NomBank frames. Fig. 3 presents the examples of semantic roles structure or predicate arguments structure (PAS).

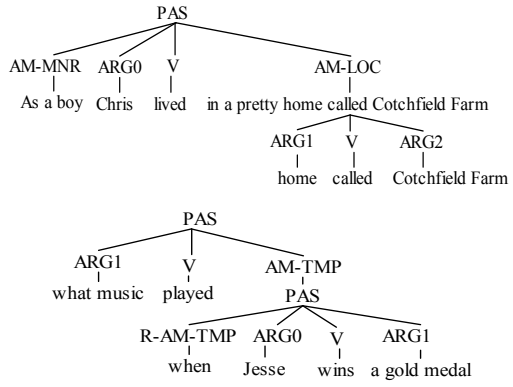

Fig.3 The semantic role structures of a sentence and a question

Research on answer extraction in QA has proved that question classification which gives the expected answer type of a question could contribute to large performance improvement for QA systems. We hence hope that it can also have positive effect in reading comprehension tests.

The questions answered in this paper start with question word *who*, *when*, *where*, and *what*. We perform rule-based question classification to assign expected answer type to each question. The rules are given in Appendix.

For *who* questions, the expected answer types include human description tagged as **HUM_DEF**, human as **HUM**, organization as **ORG**, and human or organization as **HUM_ORG** meaning the answer might be either a human name or an organization.

For *where* questions, the expected answer type will be assigned as **LOC** expressing the answer should be a location. The answer type will be assigned as **DTIME** for *when* questions and their answers will be either date or time.

What questions are very diverse in their answer types including event, activity, numeric such as date and code, entity, and description, etc. We first define the category **EVENT**, **ACT**, **DEF**, **DESC** for those questions whose answers cannot be a physical entity and should be a kind of event, activity, definition or description, and **ENTITY** for those answered with a physical entity name, **OTHER** for those whose actual categories are unknown. Furthermore, we append fine category for coarse ENTITY category. These fine categories named as **ENTITY_*noun*** are dynamically determined during classification.

C. Conceptual graph building

We build the conceptual graph for every sentence in the text, and then supplement the omitted facts about the events or entities in it with relations in other sentence graphs which are about the same entity or event. All conceptual graphs are built relying on grammatical dependency relations and semantic roles.

1) Building the conceptual graph for a sentence or question

The graph for a sentence or question is built through the following steps:

Step 1 – *Identify the verbal and nominal predicates in the sentence or question*

Step 2 – *Build the conceptual graph representation for each of the semantic role constituents of every predicate.*

The semantic role constituent might be a phrase, or a clause. Phrases or clauses are likely to contain predicates and embedded semantic roles, thus the building is done recursively by analyzing the grammatical and semantic role structure in the phrase or clause.

Step 3 – *Link all the conceptual graphs of every semantic role to a single graph.*

All the conceptual graphs built for semantic roles in the previous step are attached to their corresponding

1. http://nlp.cs.lth.se

predicate concept nodes, to form the conceptual graph representation for the complete sentence.

A problem is that no semantic role is labeled for the verb "*be*". For the sake of representation consistency we regard its subject as role "ARG0". Its object and other modifiers are regarded as "AM-TMP", "AM-LOC" or "AM-EXT" according to the type of the head words in the modifiers.

The syntactic constituents corresponding to a semantic role can be a noun phrase, prepositional phrase, subordinate clause, adjective phrase, adverb, or modal verb. The graph representation of each type is described in the following.

(1) *Noun phrase*

The simple syntactic patterns of noun phrases are NP -> DT NN, NP-> CD NNS, NP -> PRP, and NP->NNP. Other more complex patterns can be formed if attaching some modifiers for the head of the phrase. The compound of several coordinate noun phrases will be separated into several single noun phrases, and the conceptual graph of each of these phrases will be built and then linked to the main graph of the sentence. The building process for a noun phrase is as follows:

Step 1– *Recognize the head of the phrase.*

If the head is a part of a named entity, extend it to be the complete named entity.

Step 2– *Create the concept node for the head*

If the head is a named entity or personal pronoun, the type of the concept will be the name of entity type or PERSON, and the referent of the concept is the named entity or pronoun; otherwise, the type will be the head itself, and the referent is the determiner or a blank. For question sentences, the head might be an interrogative or modified by an interrogative (e.g. "what music"). If so the type of the concept should be the question category and the referent will be a special string "ANS".

Step 3– *Create the conceptual graphs for modifiers of the head*.

For modifiers which might be noun phrases, adjective phrases, preposition phrases, subordinate clauses, create the conceptual graphs for them and then link these graphs to the concept of the head of the noun phrase with the relation type "attr". For modifiers in the format of gerunds or past participles, in which there will be locally labeled predicates and their semantic roles, the conceptual graphs for them would be built separately and the head of the noun phrase will be a node in these graphs, thus these graphs will be naturally linked with the main graph of the sentence.

(2) *Prepositional phrase and subordinate clause*

In most cases, a prepositional phrase consists of a preposition and a noun phrase. We first construct a conceptual graph representing the noun phrase following the transformation rules introduced before, and then use the preposition as the relation to link the graph of the noun phrase and the main graph of the sentence. For subordinate clauses, we use the method of Hensman (2004, 2005) to construct and link their conceptual graphs.

(3) *Adjective phrase*

The syntactic patterns of adjective phrases include "ADJP->JJ PP", "ADJP ->JJ TO VP", "ADJP -> JJ SBAR", "ADJP->JJ [CC JJ]*". We first build the concept to represent the adjective using the adjective itself as the type and blank as the referent. If the phrase is composed of several coordinate adjectives connected with conjunctions, these adjectives will be transformed into several separated concepts corresponding to each of them. Otherwise, the graph for the phrase following the adjective should also be built and then linked to the concept of the adjective.

It is likely that some adjective (e.g., *three years old*) has noun phrase or adverb modifiers. In this case, its adverb or noun phrase modifiers will be regarded as attributes. After the graphs of these modifiers are built, they should be linked to the adjective concept node with the relation "attr".

An adjective phrase can be a semantic role by itself, or be a modifier of a noun phrase in a semantic role. After the conceptual graph of the adjective phrase is built, we link it to the concept of its head with the relation "attr" if it is a modifier, or link it to the concept of the predicate.

(4) *Attaching all constituents to the predicates*

The concept for a predicate should be constructed before linking the graphs of its all semantic role constituents. For nominal predicate words, we replace them with their source verb words provided in NomBank frames (e.g., id="abuse.02" source="verb-misuse.01").

After the conceptual graphs for all constituents are built, we attach these conceptual graphs to the predicate concept with the relations whose types are role names. If the constituent is a prepositional phrase heading with preposition *prep*, the relation type will be the combination of role name and *prep* with underline.

In traditional conceptual graph, the concept node doesn't contain more information than type and referent. To make the matching between question and text convenient, we append more information into concept nodes, including the part-of-speech (POS), the sentence sequence number, word sequence number in the sentence, etc.

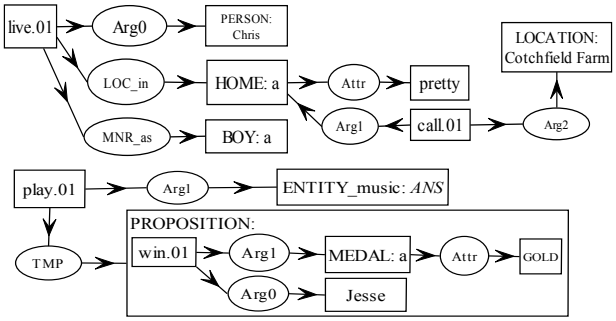

Fig. 4 Conceptual graphs of a sentence and a question

Fig. 4 shows the examples of the conceptual graphs built for the sentence "*As a boy, Chris lived in a pretty home called Cotchfield Farm.*" and the question "*What music played when Jesse wins a gold medal*". The appended POS, sentence numbers, word numbers are not given in nodes.

2) Complementing the conceptual graphs of sentences

Stories for grade school children are often used as corpus in RC tests research. The sentences in these stories are generally short and the facts about an event or entity are often scattered over multiple sentences. For example, in passage "[*The first merry-go-round in the United States was built in 1799.*]$_{S1}$…[*The merry-go-round was built in a park in Salem.*]$_{S2}$", the location and temporal information about the building of the merry-go-round are described separately in different sentences. In passage "[*At noon, two small children cut a ribbon*]$_{S1}$… [*After it was cut, people walked through the door for the first time.*]$_{S2}$", the agent and temporal role about event "*cut*" in sentence S2 are omitted.

To make the events represented in conceptual graphs complete, when there are omitted constituents in a sentence, or the facts about an event are scattered over multiple sentences, we complement the sentence graph with the nodes and relations from other sentence graphs about the same event. The following algorithm is used to merge and complement the conceptual graphs of sentences.

Algorithm: complement_concept_graph(*GS_LIST*)
Input: *GS_LIS*, list of conceptual graphs of all sentences
Output: *MERGED_GS_LIST*, list of merged conceptual graphs of all sentences

Initialize: *DICT*={} and *MERGED_GS_LIST* = [ ]
for each predicate concept *C* in graphs in *GS_LIST*:
  REL_DICT_C = {}
  for each relation *rel* with type *t* started from *C*:
    $REL\_DICT_C[t]$ <= the concept pointed by *rel*
  $DICT[C] <= REL\_DICT_C$

for each graph *GS* in *GS_LIST*
  for each predicate concept *C* in *GS*
    for each predicate concept *C** in graphs in *GS_LIST*
      $REL\_DICT_C <= DICT[C]$
      $REL\_DICT_{C*} <= DICT[C*]$
      if not same_event(*C*, REL_DICT_C, *C**, REL_DICT_{C*})
        then continue
      for each *key* and *value* in REL_DICT_{C*}
        if key not in REL_DICT_C.keys() then:
          create new node *p* with the type and referent
            from the concept *value*;
          attach *p* to *C* with *key* as the relation type
save *GS* into *MERGED_GS_LIST*

The algorithm to judge whether two predicate concepts are about the same event is as follows:

Algorithm: same_event(*C*, REL_DICT_C, *C, REL_DICT_{C*})**
Input: predicate concept C and C*, and their relations dictionary REL_DICT_C and REL_DICT_{C*}
Output: Boolean value showing whether *C* and *C** are about same event

if the type in *C* and the type in *C** are not same, then return false
if the type sense in *C* and the type sense in *C** are not same, then
  return False
flag = True
for each *key* and its *value* in REL_DICT_C:
  if key in REL_DICT_{C*}.keys() and $REL\_DICT_{C*}[key]$ <> value,
  then
    flag = false
    break
if flag then return True else return False

After merging, the conceptual graph for every sentence will contain the complete propositions about the event mentioned in the sentence. As a whole, a concept for a physical or abstract entity can be contained in multiple sentence conceptual graphs. These concept nodes will be the intersection among sentence graphs, thus all conceptual graphs of sentences are linked together logically by these entity concept nodes. The conceptual graph of a text is the union of all sentence conceptual graphs.

D. Candidate answer concept extraction and ranking

Before answer extraction, the question is classified into a specific category according to its expected answer type. These categories include ACT, EVENT, DESC, DEF, HUM, ORG, HUM_ORG, DTIME, LOC, ENTITY_*noun*, and OTHER. Different question categories require different answer string format.

For HUM_DEF and DEF questions, which are commonly expressed in patterns like "*who is* <OBJECT>?" or "*what is* [*a*] <OBJECT>?", the answers are usually presented in patterns such as "<OBJECT> *is* <DEF>", "<DEF> *is called* <OBJECT>", "<OBJECT>, <DEF>", etc. Thus we

present a heuristic method to extract the definition concept node from the conceptual graph of the text. We first identify the object concept in question graph, and then search the definition concepts in the text graph. These definition concepts and the object concept should be linked to common predicate concepts such as "*be*", "*call*", "*name*" or "*label*", with the linkages starting with the predicate concepts and ending with the object concept or the definition concept. The definition concept might be linked to the object concept with the relation type "*appo*" if it is appositive grammatical relation between them.

For other question categories, we extract their answer nodes from the text graph through the following two steps:

Step 1- *Extract the candidate answer nodes from the text graph*

For ACT and EVENT questions, the answer nodes should be verbal predicate concept (not including "be" nodes). For HUM, ORG, HUM_ORG, DTIME and LOC questions, the answer nodes should be the concepts whose types pertain respectively to "human", "organization", "human or organization", "date or time", and "location".

FOR ENTITY_*noun* questions, their answers should be physical entities. We extract the noun concepts which are for physical entities as candidate answer nodes. FOR DESC questions, all noun concepts not being physical entity, location, date, and time are selected as candidate answer nodes. For OTHER questions, the candidate nodes include all of noun concepts that are not location, date and time.

Step 2- *Rank all candidate answer nodes*

For every candidate answer node, we extract the sub-graphs by traversing the text graph in depth-first way from this node. Traversal process should finish if the already extracted sub-graph contains all of concept nodes of the question graph or the steps exceed a threshold.

While traversing the text graph from a candidate answer node, every step to extract nodes or relations should satisfy the following rules:

1) If a concept node is extracted, all relation nodes linked to by directed edges started from this concept node should also be extracted.

2) If the extracted node is a relation node, the concept node pointed to by the relation edge in another end should also be extracted.

Fig. 5 is an example of extracted sub-graph after the depth-first traversal of three steps from the candidate answer node [*place*: *a*]. In the first step, the nodes [*be*] and [*ship*: *the*] are extracted. In the second step, the nodes [*carry*], [*oil*], [*car*] and [*truck*] are extracted, and in the third step, the nodes [*pour*], [*sea*: *the*] are extracted.

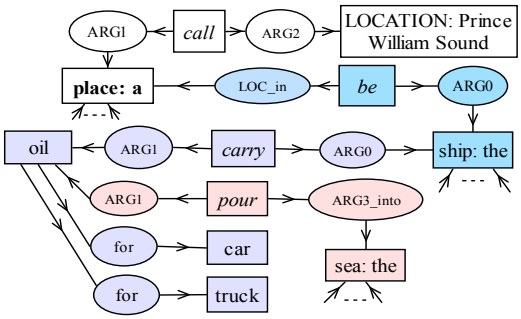

Fig. 5. A sub-graph of a location candidate answer node

The extracted sub-graph corresponding to a candidate answer node can be separated into two parts. The first part is the concept nodes set $NodeSet_c$ in this sub-graph, and the second part is the relations set $RelSet_c$, in which every relation consists of begin node, relation type, and end node. For example, the node set of the sub-graph in Fig. 6 contains [*place*:*a*], [*be*], [*ship*:*the*], [*carry*], etc. The relation set of this sub-graph contains the relations such as [*be*]->(LOC_*in*) -> [*place*: *a*], [*be*] -> (ARG0) -> [*ship*: *the*], [*carry*]-> (ARG0)-> [*ship*: *the*], etc.

After the question graph is also separated into node set $NodeSet_q$ and relation set $RelSet_q$, the matching similarity between the question graph and a sub-graph containing the candidate answer node can be calculated with the equation:

$$Sim = \alpha \cdot Sim_{nodes} + (1-\alpha) \cdot Sim_{rels}, \quad (1)$$

where Sim_{nodes} denotes the matching similarity between the node set of question graph and that of the sub-graph of candidate answer node, which can be defined as:

$$Sim_{nodes} = \frac{\sum_{node_q \in NodeSet_q} sim(node_q, node_c^*)}{n(NodeSet_q) + n(NodeSet_c)}, \quad (2)$$

in which $n(NodeSet_q)$ or $n(NodeSet_c)$ is the number of nodes in NodeSetq or NodeSetc, and $node^*_c$ is defines as:

$$node_c^* = \underset{node_c \in NodeSet_c}{\text{argmax}} \; sim(node_q, node_c), \quad (3)$$

where $sim(node_q, node_c)$ denotes the similarity between two concept nodes. Since a concept node is characterized by two elements: type and referent, the similarity between them can be defined as:

$$sim(node_1, node_2) = sim(type_1, type_2) \cdot sim(ref_1, ref_2), \quad (4)$$

here $sim(type_1, type_2)$ and $sim(ref_1, ref_2)$ are the similarities respectively between types of two nodes and between referents. $sim(type_1, type_2)$ is defined as:

$$sim(type_1, type_2) = \begin{cases} 1, \text{ if } type_1 = type_2 \\ 1, \text{ if } type_1 \text{ is coreference with } type_2 \\ \text{pathsim}_{WN}(type1, type2), \text{ else} \end{cases}, \quad (5)$$

in which $\text{pathsim}_{WN}(type1, type2)$ is the WordNet-based path similarity between two words $type_1$ and $type_2$. And $sim(ref_1, ref_2)$ is defined as:

$$sim(ref_1, ref_2) = \begin{cases} 1, \text{ if } ref_1 = ref_2 \\ 1, \text{ if } ref_1, ref_2 \in \{\text{'a','the','an',''}\} \\ 0, \text{ else} \end{cases}. \quad (6)$$

In the second term in Equation (1), Sim_{rels}, which denotes the matching similarity between the relation set of question graph and the relation set of the sub-graph of a candidate answer node, is defined as:

$$Sim_{rels} = \frac{\sum_{rel_q \in RelSet_q} sim(rel_q, rel_c^*)}{n(RelSet_q) + n(RelSet_c)}, \quad (7)$$

in which $n(RelSet_q)$ or $n(RelSet_c)$ is the number of relations in $RelSet_q$ or $RelSet_c$, and rel^*_c is defines as:

$$rel_c^* = \underset{rel_c \in RelSet_c}{\text{argmax}}\, sim(rel_q, rel_c), \quad (8)$$

where $sim(rel_q, rel_c)$ is the similarity between two relations rel_q and rel_c. A relation consists of a begin concept node, an end concept node, and a relation type, thus the similarity between two pairs of relations can be defined as:

$$sim(rel_1, rel_2) = sim(b_1, b_2) \cdot sim(e_1, e_2) \cdot sim(r_1, r_2) \quad (9)$$

in which, $sim(b_1, b_2)$ is the similarity between the begin node b_1 of rel_1 and the begin node b_2 of rel_2, while $sim(e_1, e_2)$ is the similarity between the end node e_1 and e_2 of rel_1 and rel_2, and $sim(r_1, r_2)$ is the similarity between the relation types of two relations rel_1 and rel_2.

$$sim(r_1, r_2) = \begin{cases} 1, \text{ if } r_1 = r_2 \\ \lambda_1, \text{ if rolename}(r_1) = \text{rolename}(r_2) \\ \lambda_2, \text{ else} \end{cases} \quad (10)$$

As mentioned in Subsection C of Section IV, if the semantic role is a prepositional phrase, then the relation type between it and its predicate will be the combination of the role name and preposition *prep*. In this case, the relation type similarity between two relations rel_1 and rel_2 will be λ_1 if the two types start with the same role name although they are not strictly similar.

All the extracted sub-graphs containing the candidate answer nodes will be scored according to their matching similarity with the question graph, and the maximum score will be seen as the score of the candidate answer node.

Thus every candidate answer node will receive a score after this step, and then can be ranked in descendant order according to their scores. In the ranked list of all candidate answer nodes, the node ranked first will be returned as the answer seed node to form the answer string.

E. Answer string formation

The answer string responding to the question will be extracted from the original sentences in the text. The answer node returned from the previous process will be used to form the seed word as preparation for answer string extraction. If the type of the answer node is one of PERSON, LOCATION, DATE, TIME, or NAMEDENTITY, the seed word, which should be a person name, location name, date, or time, will be the referent in the answer node. Otherwise, the type of the answer node will be seen as seed word.

When the question category is EVENT or ACT, the seed word of the answer node should be a verb or predicate and the answer string should be the complete event description involving the predicate and associated semantic roles. As discussed before, any sentence containing the seed word might not include all the semantic roles (facts) of the predicate (event). So we select the best sentence from the text as the source of the answer string. This sentence should contain the seed word and the most number of its semantic role constituents. The fragment covering the seed word and its semantic roles from this sentence will be returned as the final answer string.

For other question categories except EVENT and ACT, the earliest occurred sentence containing the seed word in the text will be selected to extract the answer string. Employing the grammatical relations in the sentence, we extract the sentence fragment covering the seed word itself and all of its modifiers. This fragment will be returned as the final answer string.

V. EVALUATIONS

A. Corpus and evaluation metric

We also adopted the Remedia corpus used by previously work as our evaluation data. Fig. 1 contains a sample story from this corpus. The corpus is divided into two sets of roughly 60 stories. One set is for traing and another one for test. Accompanying each story are five short questions: *who*, *what*, *when*, *where*, and *why*. We did not include the *why* questions in experiments since *why* questions require discourse

processing, which is beyond the capability scope of our system.

The corpus provided the teacher's answer keys for every question, but as discussed in (Wellern et al, 2005), these answer keys are insufficient. We enriched manually the answer key as appropriate, and added multiple possible answer strings.

Moreover, we also use the coreference and named entity information in the corpus annotated manually by MITRE.

We used a strict scoring metric for the responses of system to questions. If the response is in right answer sting list, then this response will be labeled as correct; or the response will be incorrect. Then we can get the accuracy of exact answer string response as follows:

$$accuracy = \frac{\#correct\ responses\ to\ questions}{\#questions} \quad (11)$$

B. Experimental results and analysis

As discussed in Subsection D of Section IV, the sub-graph for every candidate answer node should be extracted with the way of depth-first traversal on the text graph, and then to be matched with the question graph to get the ranking score for the candidate answer node. Traversal process should finish when the already extracted sub-graph contains all concept nodes of the question graph, or the traversal steps exceed a threshold. We set the threshold (i.e., the maximum steps to traverse) as three steps heuristically after observing the training data.

While matching the sub-graph of a candidate answer node with the question graph, several parameters should be determined. The first parameter α is used to weight the node set matching similarity in the linear interpolation Equation (1). The second and third parameters are λ_1 and λ_2 in Equation (10), and we set them as 0.5 (1/2) and 0.25 (1/4) respectively, which are approximate similarity score for two relation types while they are not strictly similar.

We select the value of the parameter α through experimental validation on the training data. Fig. 6 shows the performance changes of four question types on the training data given different values of the parameter α.

For *when* and *where* questions, the method achieves the highest performance when the parameter α is 0.6. But for *who* questions, the highest accuracy is attained when α=0.5, and for *what* questions, it achieves the highest accuracy when α=0.4. Thus there are performance difference among different question types while α is set as different value.

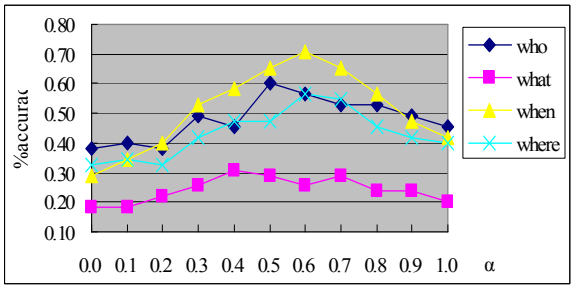

Fig. 6. The performance curves of different question types on the training data.

To select a uniform value of α for different question types, the overall performance variation should be investigated after assigning α different value. Fig. 7 shows the performance curve of overall questions on the training data at different values of α, from which we can see the trend that the best performance is achieved while α is assigned as 0.6.

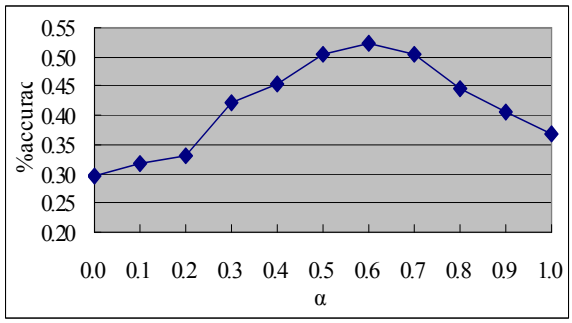

Fig. 7. The overall performance curve on the training data

Thus 0.6 will be selected as the uniform value for parameter α to fuse the concept node set similarity and the relation set similarity on the test data. Fig. 8 shows the results when we ran the system for various question types on the Remedia test data.

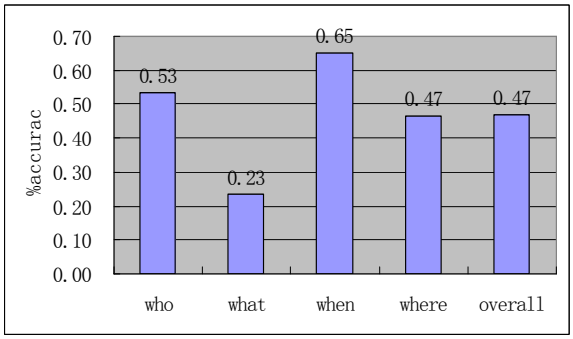

Fig. 8. Results on the test data

Answer accuracies vary dramatically among question types. The system performs best on *when* questions, and worst on *what* questions. *What* questions are particularly difficult for a number of reasons. First, our question categories designed for *what* questions are very coarse (only includes EVENT, ACT, DEF, DESC, ENTITY_*noun*, and OTHER), in which OTHER covers many unknown categories of questions, which causes that there are too many candidate answer nodes extracted from the text graph and the right answer node is more difficult to be ranked as top one. For the questions whose expected answer type is ENTITY_*noun*, there might be very less or even no candidate answer nodes to be selected because the named entities labeled in the corpus just include person name, location name, date, and time. Furthermore, for those questions whose category is DESC, the answer is a complex description instead of a noun or verb concept, so it is essentially difficult to answer. For example, the question "*What was Dr. Barry's secret?*", the answer should be "*Dr. Barry was really a woman*" and can only be correctly answered using discourse structure information.

Table 1. Comparison with other methods

type	Riloff et al. 2000	Du et al. 2005	Xu et al. 2004	Willner et al. 2005	CG matching method
who	0.41	0.48	0.54	0.52	0.53
what	0.28	0.41	0.33	0.13	0.23
when	0.55	0.55	0.62	0.45	0.65
where	0.47	0.36	0.30	0.28	0.47
overall	0.43	0.45	0.45	0.35	0.47

We also compared our results ("CG matching method" column in Table 1) across various interrogatives with previously reported results. While the methods in (Riloff et al., 2000), (Du et al., 2005), and (Xu et al., 2004) were to extract sentences as final answers and *HumSent accuracy* was used as performance metric, the method of Willner et al. (2005) was to extract concise strings as answers, in which the performance metric is similar with that used in this paper.

Although the performance metrics for these methods are not uniform, we list their results and can still compare them concerning that *HumSent accuracy* and concise answer string accuracy are similarly to evaluate the correctly answered questions portion in all questions from different points of views. From table 1, we observe that our approach in the use of conceptual graph matching method gain to RC performance overall.

We still observe that our system is unable to correctly answer 53% of the questions in Remedia test data. Some errors are caused by wrong grammatical and semantic analysis results. In other not correctly answered questions, some can only be answered using discourse structure among sentences. For example, the answer for the question "*Where did the ship sink* ?" should be extracted from the two sentences "*A terrible tragedy has happened in the middle of the ocean. A huge ship sank last night after hitting an iceberg*." using discourse coherence between them, in which "*a terrible tragedy*" refers to "*ship sank*". Some questions ask with very flexible interrogatives, which cause wrong question classifications. For example, the question "*Who is about to receive a shipment of hula hoops*?" seems to ask for a person, but the real answer is a place "*South Pole*". Furthermore, some questions need deep semantic reasoning to answer, and they hence are very difficult and even beyond the capabilities of current technologies to extract their answers.

VI. CONCLUSIONS

This paper proposes a conceptual graph matching method to extract the concise answer strings for the questions in reading comprehension tests. The method first constructs a conceptual graph for the text and the question according to the grammatical and semantic relations in the text and question. Instead of comparing every single sentence in text with the question and selecting sentence which has the largest word overlap with the question to extract answer string, the method extracts the sub-graphs from the text graph for every candidate answer node, then matches these sub-graphs with the question graph, and selects the candidate answer node, whose sub-graph has the largest matching score with the question graph, as the seed node to form and extract the final answer string. Concerning the problem that the propositions mentioned in the question might be scattered over multiple text sentences, or the answer is not in the sentence which has the largest word overlap with the question, our method improves the system performance due to that the sub-graphs of every candidate answer node is not restricted on a single sentence.

In the near future, our work will focus on the better matching algorithm between the sub-graph of candidate answer nodes and the conceptual graph of the question, including the selection of parameters, and utilizing the frame set in PropBank to match the different verbs.

APPENDIX

Table 2. The rules for question classification

type	rule description
who	if subject is "*who*", predicate is "*be*": if object is a named entity then class is HUM_DESC else if object is a noun and "*person*" is in noun's hypernym then class is HUM else if object is a noun and "*organization*" is in noun's hypernym then class is ORG else class is HUM_ORG.
where	class is LOC
when	class is DTIME.
what	If "*what*" is subject of predicate "*happen*", then class is EVENT
what	If "*what*" is object of predicate "*do*", then class is ACT
what	If "*what*" is subject of predicate "*be*", object of "*be*" is a noun with the determiner "*a*", the noun has not adjective modifiers, then class is DEF
what	If "*what*" is subject of predicate "*be*", object of "*be*" is a named entity, then class is DEF
what	If "*what*" is object of some predicate (not "*be*"), then class is OTHER
what	If "*what*" is modifier or determiner of a noun: If "*physical entity*" is in the noun's hypernyms in WordNet, then class is ENTITY_noun; If "*what*" is modifier or determiner of word "*kind*"or "*type*", "*kind*" or "*type*" is modified by preposition "*of*", and "*of*" is modified by a noun: If "*physical entity*" is in the noun's hypernyms, then class is ENTITY_noun ; else class is DESC
what	If "*what*" is subject of predicate (not "*be*"), then class is OTHER
what	If "*what*" is subject of predicate "*be*", object is "*name*": if "*name*" is modified by "*for*" or "*of*" preposition phrase, and "*for*" or "*of*" is modified by a *noun*, then class is ENTITY_*noun* else if "*name*" is modified by a *noun* with possession relation, then class is ENTITY_noun
what	If "*what*" is subject of predicate "*be*", object is a noun (not "*name*"): If "*physical entity*" is in the *noun*'s hypernyms, then class is ENTITY_ *noun* ; else class is DESC
what	If class is not determined by rule above, then class is OTHER

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant Nos. 60736044, 60675034.

REFERENCES

[1] Eugene Charniak, Yasemin Altun, Rodrigo de Salvo Braz. Reading Comprehension Programs in a Statistical-Language-Processing Class. In Proceedings of the ANLP/NAACL 2000 Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding System, 2000, pp. 1-5.

[2] Charles L. A. Clarke, Gordon V. Cormack, Thomas R. Lynam. Exploiting Redundancy in Question Answering. In Proceedings of the 24th ACM Conference on Research and Development in Information Retrieval (SIGIR-2001), 2001, pp. 358-365.

[3] Yongping Du, Helen Meng, Xuanjing Huang, Lide Wu. The Use of Metadata, Web-derived Answer Patterns and Passage Context to Improve Reading Comprehension Performance. Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), 2005, pp. 604-611.

[4] Svetlana Hensman. Construction of Conceptual Graph representation of texts. In Proceedings of Student Research Workshop at HLT-NAACL, Boston, 2004, pp. 49-54.

[5] Svetlana Hensman, John Dunnion. Representing semantics of texts – a non-statistical approach. Second International Joint Conference on Natural Language Processing: Companion Volume including Posters/Demos and tutorial abstracts, 2005.

[6] Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. Deep Read: A Reading Comprehension System. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, 1999, pp. 325-332.

[7] Marc Light, Gideon S. Mann, Ellen Riloff, Eric Breck. Analysis for elucidating current question answering technology, Natural Language Engineering, 2001, 7(4), pp. 325-342.

[8] Christiane Fellbaum, editor. WordNet: An Electronic Lexical Database. MIT Press, May 1998.

[9] Hwee Tou Ng, Leong Hwee Teo, Jennifer Lai Pheng Kwan. A Machine Learning Approach to Answering Questions for Reading Comprehension Tests. In Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 2000, pp. 124-132.

[10] Lehnert, W. A conceptual theory of question answering. IN: Grosz, B. J., Jones, K. S. and Webber, B. L.,editors, Reading in Natural Language Processing. Morgan Kaufmann, 1986, pp. 651-658.

[11] Ellen Riloff and Michael Thelen. A Rule-based Question Answering System for Reading Comprehension Tests [A]. In Proceedings of ANLP/NAACL 2000 Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems, 2000, pp. 13-19.

[12] John F. Sowa. Semantics of conceptual graphs. Proceedings of the 17th annual meeting on Association for Computational Linguistics, 1979, pp. 39-44.
[13] John F. Sowa and Eileen C. Way. Implementing a semantic interpreter using conceptual graphs. IBM Journal of Research and Development, 30(1), pp. 57–69, January, 1986.
[14] Ben Wellner, Lisa Ferro, Warren Greiff and Lynette Hirschman. Reading comprehension tests for computer-based understanding evaluation. Natural Language Engineering, 2005, pp.1-30.
[15] Kui Xu and Helen Meng. Using Verb Dependency Matching in a Reading Comprehension System [A]. First Asia Information Retrieval Symposium (AIRS 2004), 2004. pp. 190-201.
[16] Kui Xu, Helen Meng, Fuliang Weng. A Maximum Entropy Framework that Integrates Word Dependencies and Grammatical Relations for Reading Comprehension. In Proceedings of the Human Language Technology Conference of the NAACL, 2006, pp 185-188.

Zhi-Chang Zhang
Born in 1976. He is currently a Ph.D. candidate of Dept. computer science and technology at Harbin Institute of Technology. He received the B.S. degree in computer and application from Northwest Normal University, Lanzhou, China in 1998 and the M.S. degree in computer software and theory from Northwestern Polytechnical University, Xi'an, China in 2003. His research interests are in natural language processing and information retrieval. Email: pangzhang@gmail.com.

Yu Zhang
Born in 1972. He received the B.S., M.S. and Ph.D. degrees in computer application from Harbin Institute Technology, Harbin, China in 1993, 1996 and 2000 respectively. He is an associate professor of Dept. computer science and technology at Harbin Institute of Technology. His research interests are in information retrieval. Email: pangzhang@gmail.com

Ting Liu
Born in 1972. He is a professor of Dept. computer science and technology at Harbin Institute of Technology. He received the B.S., M.S. and Ph.D. degrees in computer application from Harbin Institute Technology, Harbin, China in 1993, 1995 and 1998 respectively. His research interests are in natural language processing, information retrieval and text mining. Email: tliu@ir.hit.edu.cn

Sheng Li
Born in 1943. He is a professor of Dept. computer science and technology at Harbin Institute of Technology. He received the B.S. degree in computer application from Harbin Institute Technology, Harbin, China in 1965. His research interests include natural language processing, information retrieval and machine translation. Email: lisheng@hit.edu.cn